

IN THE CLAIMS

1. (Currently Amended) A method for processing timer events, the method comprising:

receiving a timer subscription containing a time value, a timer identity and an identity of a module to notify upon expiration of the time value, each module operable to include a plurality of threads;

establishing a timer in a component execution environment to track expiration of the time value;

detecting expiration of the timer;

in response to detecting expiration of the timer, determining if the module is disabled, and if the module is disabled, enabling the module, enabling modules corresponding to activation of a corresponding component by an activation mechanism, disabling corresponding to deactivation of the corresponding component by the activation mechanism, the activation and deactivation operations operable to reduce memory consumption by inactive components and provide selective invocation to maintain availability of the component;

notifying a subscriber in the module of expiration of the timer, enabling and disabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component, and

receiving a second timer subscription to the same timer as the timer subscription, the timer identified by a timer name provided by both the timer subscription and the second timer subscription.

2. (Original) The method of claim 1 wherein the module includes a timer handler in the subscriber, the timer subscription further indicative of the timer handler, and notifying the subscriber of the expiration of the timer further comprises invoking the indicated timer handler for execution.

-3-

3. (Previously Presented) The method of claim 2 wherein establishing the timer further comprises:

adding the identity of the module to a global timer map, the global timer map operable to indicate a plurality of modules; and

adding a reference to the subscriber including the timer handler, into a local timer map associated with the module.

4. (Original) The method of claim 3 wherein invoking further comprises:

indexing, via the local timer map, a dispatch command operable to dispatch the timer handler.

5. (Original) The method of claim 4 wherein the local timer map includes an entry indicative of the subscriber including the timer handler within a module and the global timer map includes an entry indicative of the module.

6. (Original) The method of claim 4 wherein the reference is a dynamic offset from a base to the location in a particular instantiation of the module, the base operable to change upon reenabling of the module.

7. (Original) The method of claim 2 wherein the expiration of the timer and resulting timer initiated invocation of the timer handler is independent of the enabling of the subscriber including the timer handler.

8. (Original) The method of claim 4 wherein determining if the module is disabled further comprises:

employing the global timer map to find the entry corresponding to the timer expiration to determine the identity of the module corresponding to the timer event; and

determining, from the identity of the module, if the module is disabled.

9. (Original) The method of claim 1 wherein the timer subscription is operable to indicate periodic and aperiodic expiration times.
10. (Previously Presented) The method of claim 2 wherein the subscription is received from a subscriber within the module, the subscriber including the timer handler.
11. (Original) The method of claim 1 wherein receiving the subscription further comprises receiving a subscription from multiple subscribers in the module, each subscriber operative to include a timer handler, further comprising, in response to detecting expiration of the timer, enabling disabled modules upon expiration of a timer subscribed to by any of the multiple subscribers.
12. (Canceled)
13. (Original) The method of claim 1 further comprising resetting the expiration time value with an expiration time value from a second subscription for the same timer.
- Claims 14-15. (Canceled)
16. (Previously Presented) The method of claim 3 wherein activation and deactivation further comprises identifying, in a module server in communication with each of the modules, when to activate and deactivate modules based on information in the global timer map in a component server.
17. (Previously Presented) The method of claim 1 wherein disabling is performed by a thread manager operable to gracefully terminate each of the threads prior to deactivation, deactivation occurring by informing each of the

threads of the termination and computing when each thread has attained a termination point.

18. (Previously Presented) The method of claim 2 further comprising:
 associating the timer with a generation counter, the generation counter incrementally labeling each invocation from a particular subscriber;
 comparing, upon completion of a timer handler, the generation counter;
 canceling, if the generation counter indicates that the timer handler corresponds to the generation counter, the timer; and
 maintaining, if the timer is periodic, the pending timer corresponding to the subscriber.

19. (Previously Presented) The method of claim 1 wherein associating the timer identity with a timer handler occurs in a native language of the timer handler and corresponding subscriber, and avoids a corresponding definition in an external interface language, the external interface language for generating timer specific code.

20. (Previously Presented) The method of claim 19 wherein the external interface language is an Interface Definition Language .

21. (Currently Amended) A method for time based invocation of subscribers comprising:
 receiving a subscription to a timer, the timer associative of a timer handler and an expiration time, the timer handler having instructions operative for executing and completing a particular task and the expiration time indicative of performance of the particular task, the subscription including a timer identity;
 associating the timer with the timer handler, the association including a generic timer reference applicable to a plurality of timer handlers, the association

-6-

operable to selectively enable a module including the timer handler upon expiration of the timer handler.

receiving an indication of expiration of the timer;

determining, via the association, the corresponding timer handler and the module including the timer handler, the module operable to include a plurality of threads;

selectively enabling the module including the timer handler, enabling modules corresponding to activation of a corresponding component by an activation mechanism in a component execution environment;

dispatching the timer handler to execute and complete the time based task, enabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component, and

receiving a second subscription to the same timer as the subscription, the timer identified by a timer name provided by both the subscription and the second subscription.

22. (Previously Presented) The method of claim 21 wherein, following selectively enabling:

enqueueing an indication of the timer expiration in a queue, the queue corresponding to a process including the module containing the subscriber; and

assigning, to a particular thread corresponding to the queue, performance of the timer handler corresponding to the expired timer.

23. (Original) The method of claim 21 further comprising

associating timer event data with each timer, and delivering the timer event data to the handler upon invocation of the timer handler; and

-7-

communicating, via the association of the timer handler and the timer, the timer data, the communicating of the data independent of the location of the timer handler.

24. (Currently Amended) An infrastructure server including a computer readable storage medium comprising executable instructions encoded thereon operable to perform processing that:

instructs a module server to receive a timer subscription containing a time value, a timer identity and an identity of a module to notify upon expiration of the time value, the module operable to include a plurality of threads;

instructs a timer service in the module server to establish a timer in a component execution environment to track expiration of the time value, the timer service further operable to detect expiration of the timer, the timer service further operable to, in response to detecting expiration of the timer, determine if the module is disabled, and if the module is disabled, enable the module, wherein enabling the module corresponds to activation of a corresponding component by an activation mechanism, wherein disabling corresponds to deactivation of the corresponding component by the activation mechanism, the activation and deactivation operations operable to reduce memory consumption by inactive components and provide selective invocation to maintain availability of the component;

instructs notification of a subscriber in the module of expiration of the timer, enabling and disabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component, and

instructs the module server to receive a second timer subscription to the same timer as the timer subscription, the timer identified by a timer name provided by both the timer subscription and the second timer subscription.

25. (Original) The infrastructure server of claim 24 wherein the module includes a timer handler in the subscriber, the timer subscription further indicative of the timer handler, and the timer service further operable to notify the subscriber of the expiration of the timer.

26. (Previously Presented) The infrastructure server of claim 25 further comprising a local timer map, the timer service operable to invoke the subscriber by indexing, via the local timer map, a dispatch command operable to dispatch the timer handler.

27. (Original) The infrastructure server of claim 25 further comprising a global timer map, wherein the timer service is operable to establish the timer by:
adding the identity of the module to the global timer map, the global timer map operable to indicate a plurality of modules; and
adding a reference to the subscriber including the timer handler into a local timer map associated with the module.

28. (Original) The infrastructure server of claim 27 wherein the local timer map includes an entry indicative of the subscriber including the timer handler within a module and the global timer map includes an entry indicative of the module.

29. (Original) The infrastructure server of claim 27 wherein the reference is a dynamic offset from a base to the location in a particular instantiation of the module, the base operable to change upon reenabling of the module.

30. (Original) The infrastructure server of claim 25 wherein the timer service is operable to invoke the subscriber and the corresponding timer handler is independently of the enabling of the subscriber including the timer handler.

31. (Previously Presented) The infrastructure server of claim 25 wherein the timer service is operable to receive a subscription from a subscriber within the module, the subscriber including the timer handler.

32-33. (Canceled)

34. (Previously Presented) The infrastructure server of claim 27 wherein the module server is in communication with each of the modules, and further operable to determine when to activate and deactivate modules based on information in the global timer map in module server.

35. (Previously Presented) The infrastructure server of claim 24 wherein disabling is performed by a thread manager operable to gracefully terminate each of the threads prior to deactivation, deactivation occurring by informing each of the threads of the termination and computing when each thread has attained a termination point.

36. (Original) The infrastructure server of claim 24 further comprising, for each timer, a generation counter, the timer service operable to associate each timer with the generation counter, the generation counter incrementally labeling each invocation from a particular subscriber, the timer service further operable to compare, upon completion of a timer handler, the generation counter; cancel, if the generation counter indicates that the timer handler corresponds to the generation counter, the timer, and maintain, if the timer is periodic, the pending timer corresponding to the subscriber.

37. (Currently Amended) A computer program product having a computer readable storage medium operable to store computer program logic embodied in computer program code encoded thereon for processing timer events in a services infrastructure, comprising:

-10-

computer program code for receiving a timer subscription containing a time value , a timer identity, and an identity of a module to notify upon expiration of the time value, the module operable to include a plurality of threads;

computer program code for establishing a timer in a component execution environment to track expiration of the time value;

computer program code for detecting expiration of the timer;

computer program code for, in response to detecting expiration of the timer, determining if the module is disabled, and if the module is disabled, enabling the module, enabling modules corresponding to activation of a corresponding component by an activation mechanism, disabling corresponding to deactivation of the corresponding component by the activation mechanism, the activation and deactivation operations operable to reduce memory consumption by inactive components and provide selective invocation to maintain availability of the component;

computer program code for notifying a subscriber in the module of expiration of the timer, enabling and disabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component, and

computer program code for receiving a second timer subscription to the same timer as the timer subscription, the timer identified by a timer name provided by both the timer subscription and the second timer subscription.

38. (Currently Amended) A computer readable storage medium having program code embodying a processor based set of instructions thereon for processing timer events in a services infrastructure, comprising:

program code for receiving a timer subscription containing a time value, a timer identity and an identity of a module to notify upon expiration of the time value, the module operable to include a plurality of threads;

program code for establishing a timer in a component execution environment to track expiration of the time value;

program code for detecting expiration of the timer, the timer under the API control of application processes;

program code for, in response to detecting expiration of the timer, determining if the module is disabled, and if the module is disabled, enabling the module, enabling modules corresponding to activation of a corresponding component by an activation mechanism, disabling corresponding to deactivation of the corresponding component by the activation mechanism, the activation and deactivation operations operable to reduce memory consumption by inactive components and provide selective invocation to maintain availability of the component;

program code for notifying a subscriber in the module of expiration of the timer, enabling and disabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component; and

program code for receiving a second timer subscription to the same timer as the timer subscription, the timer identified by a timer name provided by both the timer subscription and the second timer subscription.

39. (Canceled)

40. (Currently Amended) A method for processing timer events, the method comprising:

receiving a first timer subscription containing a first time value, a timer identity and an identity of a first multi-thread module to notify upon expiration of the first time value;

-12-

receiving a second timer subscription containing a second time value, the timer identity and an identity of a second multi-thread module to notify upon expiration of the second time value;

establishing a timer in a component execution environment to track expiration of the first time value and the second time value;

detecting expiration of the timer relative to the first time value and the second time value;

in response to detecting expiration of the timer, for at least one of the first multi-thread module and the second multi-thread module:

determining if the module is disabled, and if the module is disabled, enabling the module, enabling modules corresponding to activation of a corresponding component by an activation mechanism, disabling corresponding to deactivation of the corresponding component by the activation mechanism, the activation and deactivation operations operable to reduce memory consumption by inactive components and provide selective invocation to maintain availability of the component; and

notifying a subscriber in the module of expiration of the timer, enabling and disabling being performed at a level of granularity of the modules, each of the modules corresponding to a component and operable be enabled and disabled by activation and deactivation of the corresponding component.

41. (New) The method of claim 1, wherein the component execution environment is a Common Object Request Broker Architecture (CORBA) environment, enabling further comprising loading Dynamic Link Library (DLL) for runtime binding of the activated module.

42. (New) The method of claim 41 further comprising:

searching for the module in the global timer table to locate the corresponding module;

-13-

loading, if the located module is not already loaded, the module containing a timer handler corresponding to the expired timer; and

locating the corresponding timer handler in the loaded module from the local timer table.

43. (New) The method of claim 42 wherein loading further comprises transferring the module from an unexecutable memory location to an executable memory location, loading and deactivating such that deactivating frees execution memory for use by other modules.